Building traffic matrices to support peering decisions

Paolo Lucente
pmacct

AfPIF 2017, Abidjan – Aug 2017

whoami

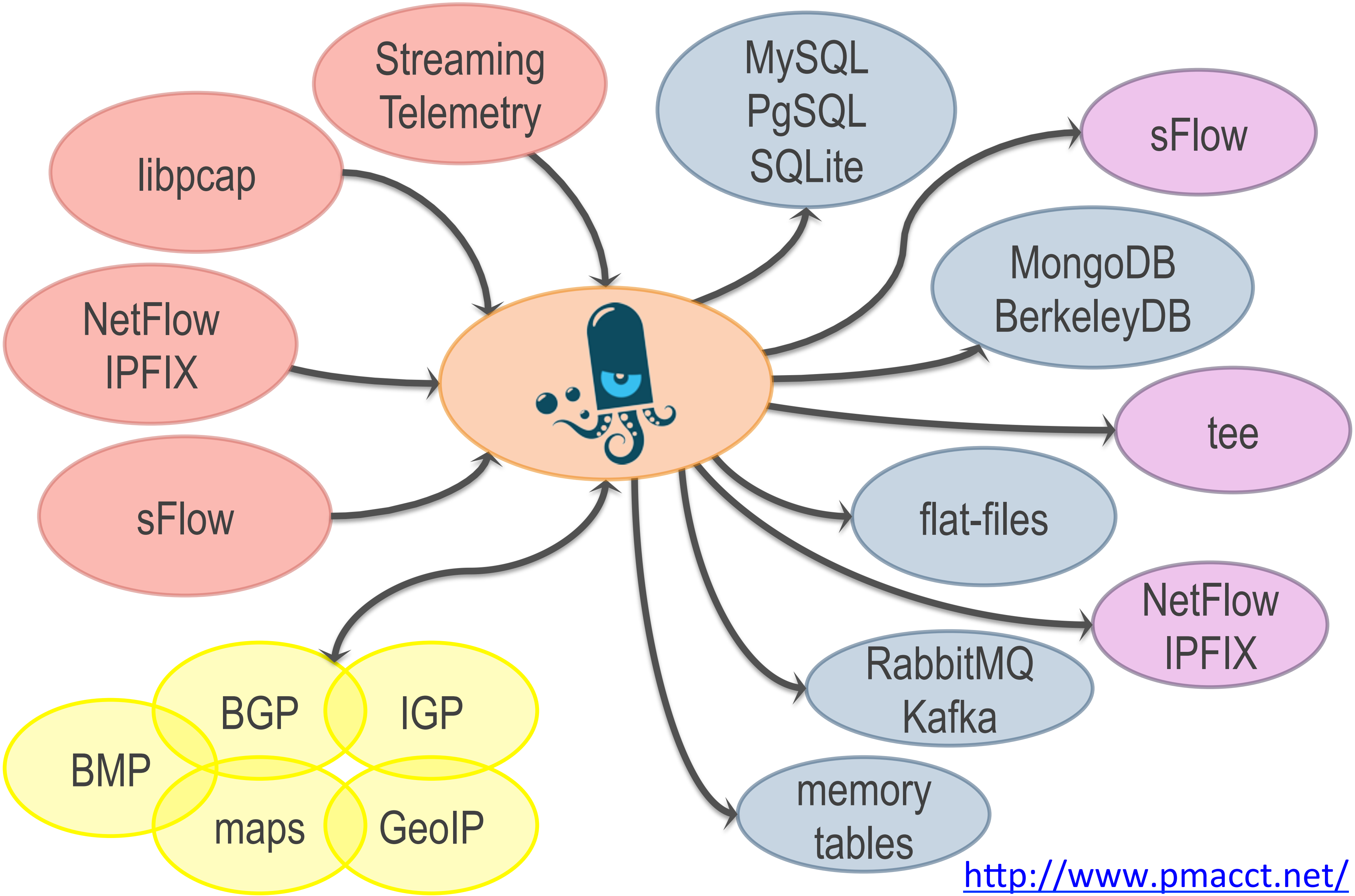
Paolo Lucente

GitHub: paololucente

LinkedIn: plucente

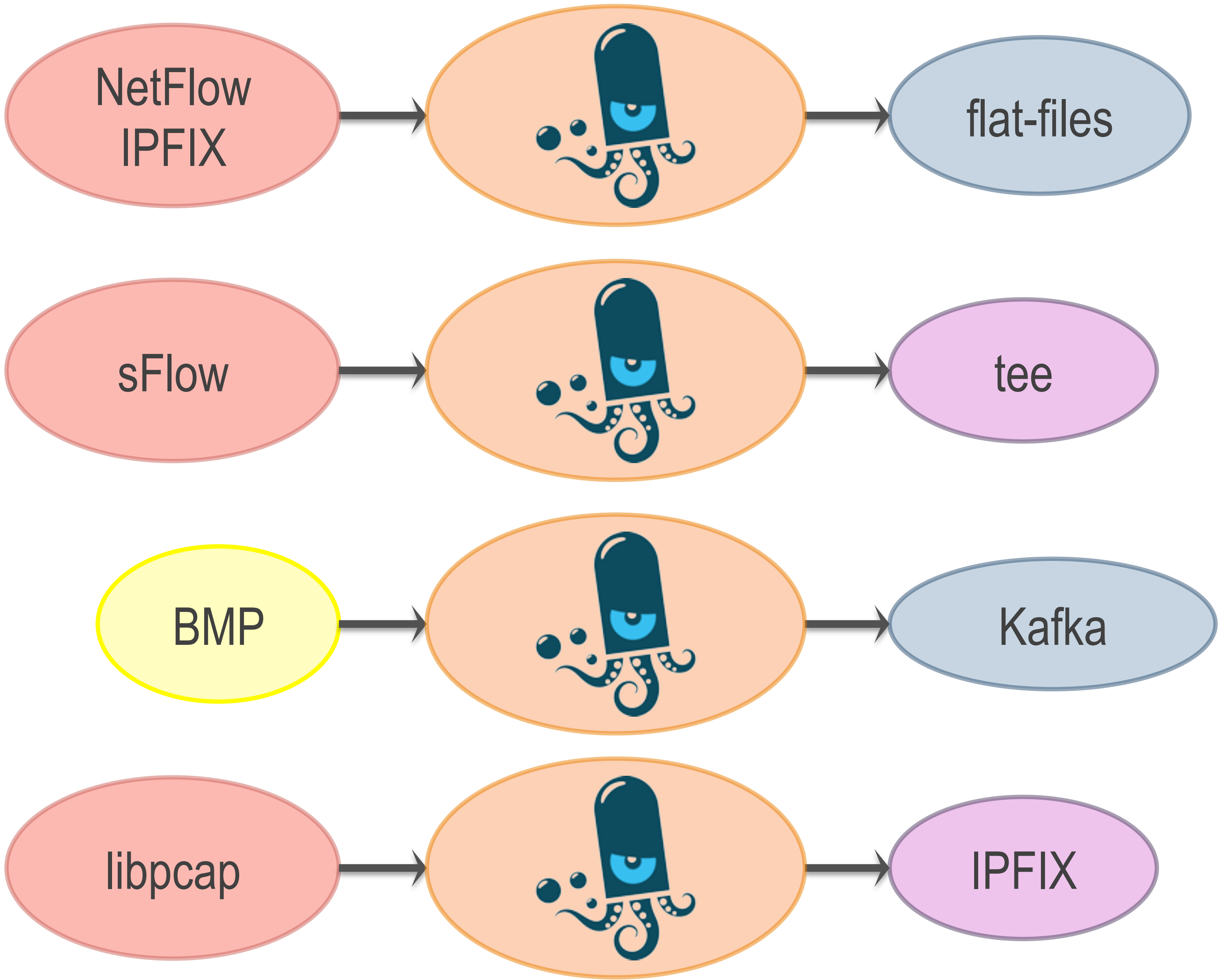
Digging data out of networks worldwide for fun and profit for more than 10 years

pmacct is open-source, free, GPL'ed software

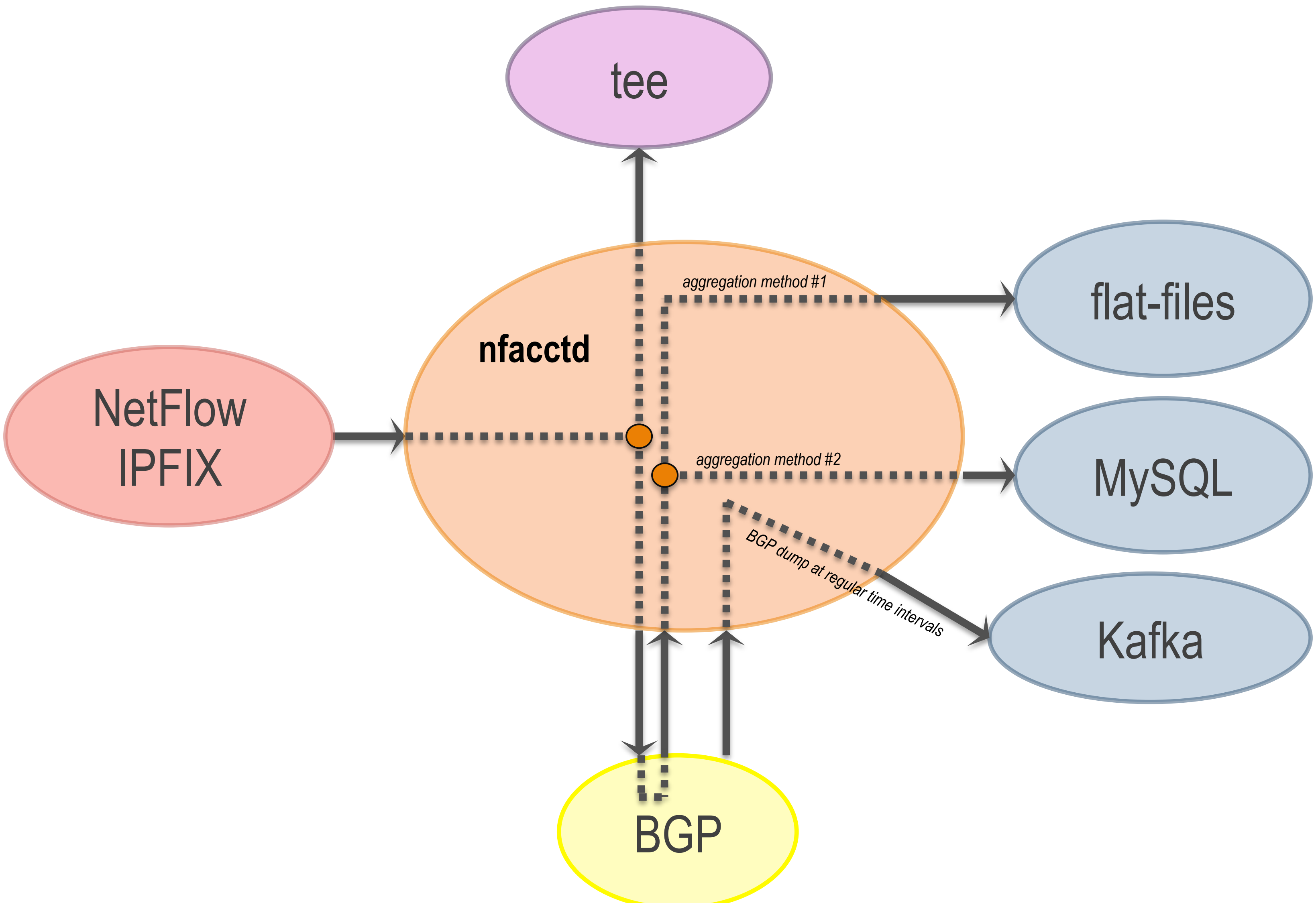


http://www.pmacct.net/

pmacct: a few simple use-cases

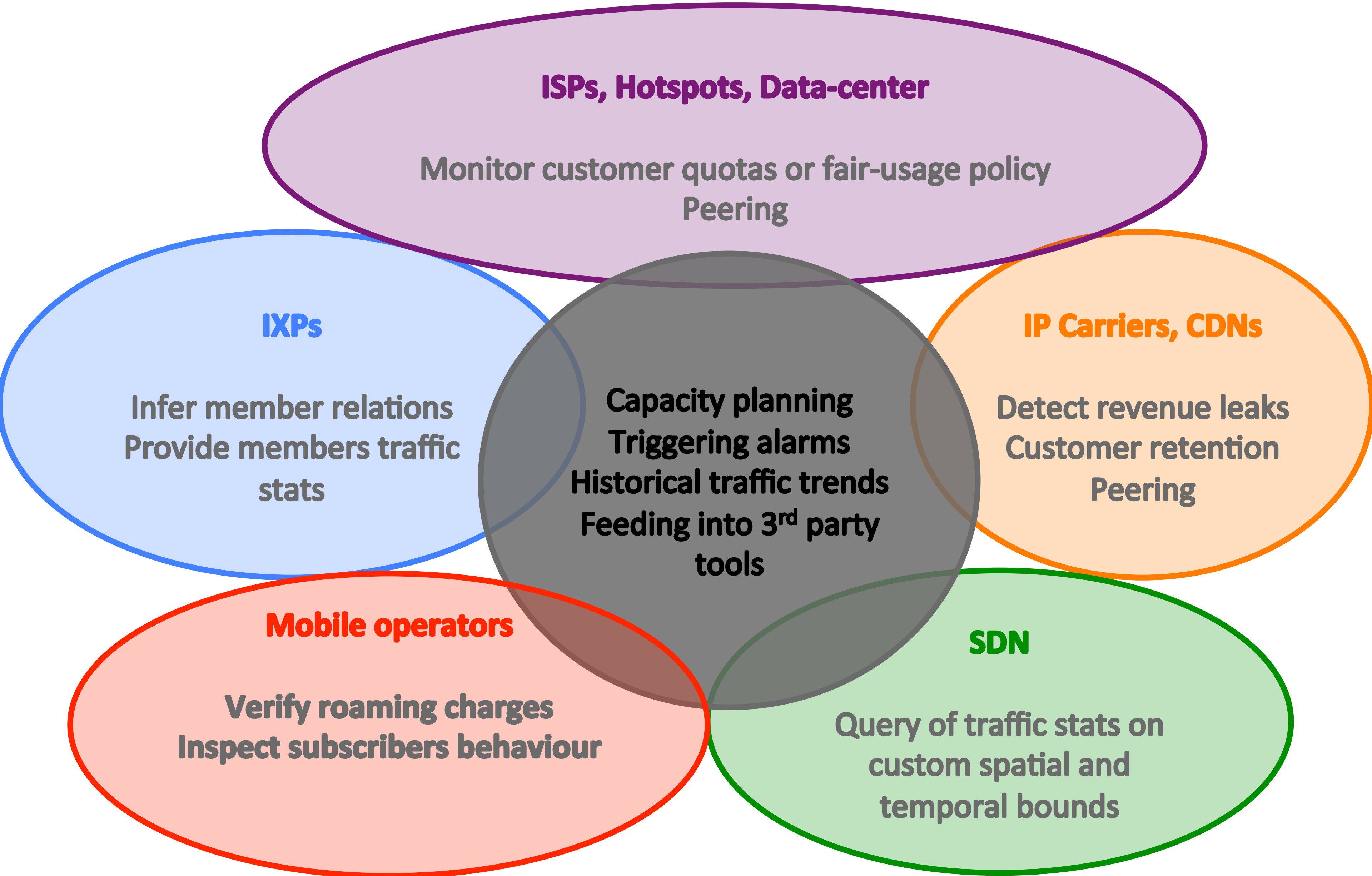


pmacct: a slightly more complex use-case



The use-case for message brokers

Use cases by industry



Key pmacct non-technical facts

- 10+ years old project
- Can’t spell the name after the second drink
- Free, open-source, independent
- Under active development
- Innovation being introduced
- Well deployed around, also in large SPs/IXPs
- Close to the SP/IXP community needs

Some technical facts (1/2)

- Pluggable architecture:
  - Can easily add support for new data sources and backends
- Correlation of data sources:
  - Natively supported data sources (ie. flow telemetry, BGP, BMP, IGP, Streaming Telemetry)
  - Enrich with external data sources via tags and labels
- Enable analytics against each data source:
  - Stream real-time
  - Dump at regular time intervals (possible state compression)

Some technical facts (2/2)

- Build multiple views out of the very same collected network traffic, ie.:
  - Unaggregated to flat-files for security and forensics; or to message brokers (RabbitMQ, Kafka) for Big Data
  - Aggregated as [<ingress router>, <ingress interface>, <BGP next-hop>, <peer destination ASN>] and sent to a SQL DB to build an internal traffic matrix for capacity planning purposes
- Pervasive data-reduction techniques, ie.:
  - Data aggregation
  - Filtering
  - Sampling

Why speaking of traffic matrices?

- Are traffic matrices useful to a network operator in the first place? Yes …
  - Capacity planning (build capacity where needed)
  - Traffic Engineering (steer traffic where capacity is available)
  - Better understand traffic patterns (what to expect, without a crystal ball)
  - Support peering decisions (traffic insight, traffic engineering at the border, support what if scenarios)

Why speaking of traffic matrices?

- Traffic matrices keep a relatively behind the scenes topic
- Some works approach the topic formally
- Other works say about the goodies of traffic matrices:
  - But where to start building one?
  - What challenges does the task present?
  - What resources do I need?
  - Which choices and options do I have?

Back to square 1

(Building traffic matrices to support peering decisions)

- What is needed:
  - BGP
  - Telemetry data: NetFlow/IPFIX, sFlow, libpcap
  - Collector: tool (ie. pmacct ☺)
  - Storage: noSQL, RDBMS or legacy (ie. RRD) solution
  - Enrichment and post-processing scripts
  - UI
- Risks:
  - 800 pound gorilla project

Getting BGP to the collector

- Needed for technical reasons:
  - Flow exporters use NetFlow v5, ie. no BGP next-hop
  - Flow exporters are unaware of BGP
  - Libpcap is used to collect traffic data
- Needed for topology or traffic related reasons:
  - Transiting traffic to 3rd parties
  - Dominated by outbound traffic

Getting BGP to the collector (cont.d)

- Let pmacct collector BGP peer with all PE devices: facing peers, transit and customers
  - No best-path computation at the collector: scalability preferred to memory usage
  - Count some 50MB of memory per full-routing table
- Set the collector as iBGP peer at the PE devices:
  - Configure it as a RR client
  - Collector acts as iBGP peer across (sub-)AS boundaries
- BGP next-hop has to represent the egress point (node or interface) of the network

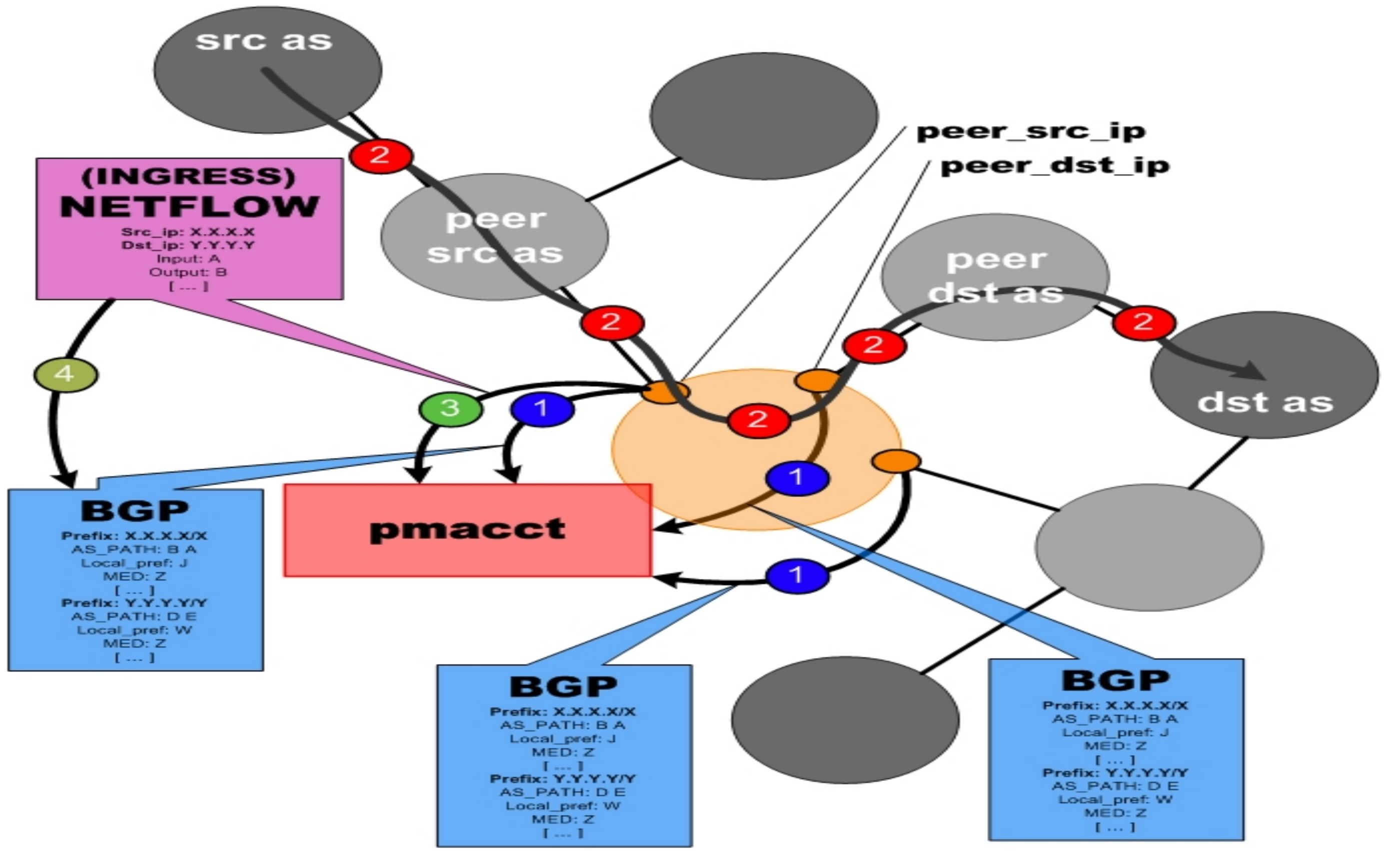
Getting telemetry to the collector

- Export ingress-only measurements at all PE devices: facing peers, transit and customers.
  - Traffic is routed to destination, so plenty of information on where it's going to
    - True, some eBGP multi-path scenarios may get challenging
  - It's crucial instead to get as much as possible about where traffic is coming from, ie.:
    - input interface at ingress router
    - source MAC address
- Perform data reduction at the PE (ie. sampling)

Getting telemetry to the collector (cont.d)

- Multiple flow collectors can be in use, ie. for different purposes. Typical export models:
  - Single tier, unicast: PE devices perform N exports
  - Multiple tiers: PEs perform export to transparent replicators in active/standby fashion; these in turn stream telemetry data to the actual collectors
- It's crucial flow collectors can tag, aggregate, filter, etc. telemetry data:
  - … might be not all data is for every collector

Telemetry data/BGP correlation



1 Edge routers send full BGP tables to pmacct

2 Traffic flows

3 NetFlow records are sent to pmacct

4 pmacct looks up BGP information: NF src addr == BGP src addr

Storing data persistently

- Data need to be aggregated both in spatial and temporal dimensions before being written down:
  - Optimal usage of system resources
  - Avoids expensive consolidation of micro-flows
- Build project-driven data set(s):
  - No shame in multiple partly overlapping data-sets
  - Optimize computing

Storing data persistently (cont.d)

- “noSQL” databases (Big Data ☺):
  - Able to handle large time-series data-sets
  - Meaningful subset of SQL query language
  - Innovative storage and indexing engines
  - Scalable: clustering, spatial and temporal partitioning
  - UI-ready: ie. ELK and TICK stacks
- Open-source RDBMS:
  - Able to handle large data-sets
  - Flexible and standardized SQL query language
  - Solid storage and indexing engines
  - Scalable: clustering, spatial and temporal partitioning

Enriching data

Tag / BGP Fields / Counters / Time

```
create table acct_bgp (
    agent_id INT(4) UNSIGNED NOT NULL,
    as_src INT(4) UNSIGNED NOT NULL,
    as_dst INT(4) UNSIGNED NOT NULL,
    peer_as_src INT(4) UNSIGNED NOT NULL,
    peer_as_dst INT(4) UNSIGNED NOT NULL,
    peer_ip_src CHAR(15) NOT NULL,
    peer_ip_dst CHAR(15) NOT NULL,
    comms CHAR(24) NOT NULL,
    as_path CHAR(21) NOT NULL,
    local_pref INT(4) UNSIGNED NOT NULL,
    med INT(4) UNSIGNED NOT NULL,
    packets INT UNSIGNED NOT NULL,
    bytes BIGINT UNSIGNED NOT NULL,
    stamp_inserted DATETIME NOT NULL,
    stamp_updated DATETIME,
    PRIMARY KEY (…)
);
```

```
shell> cat pretag.map
id=100  peer_src_as=<customer>
id=80   peer_src_as=<peer>
id=50   peer_src_as=<IP transit>
[ … ]
```

```
shell> cat peers.map
id=65534 ip=X in=A
id=65533 ip=Y in=B src_mac=J
id=65532 ip=Z in=C bgp_nexthop=W
[ … ]
```

Post-processing and reporting

– Traffic delivered to a BGP peer, per location:

```
mysql> SELECT  peer_as_dst, peer_ip_dst, SUM(bytes), stamp_inserted \
       FROM acct_bgp \
       WHERE peer_as_dst = <peer | customer | IP transit> AND
             stamp_inserted = < today | last hour | last 5 mins > \
       GROUP BY peer_as_dst, peer_ip_dst
```

– Aggregate AS PATHs to the second hop:

```
mysql> SELECT SUBSTRING_INDEX(as_path, '.', 2) AS as_path, bytes \
       FROM acct_bgp \
       WHERE local_pref = < IP transit pref> AND
             stamp_inserted = < today | yesterday | last week > \
       GROUP BY SUBSTRING_INDEX(as_path, '.', 2)
       ORDER BY SUM(bytes)
```

– Focus peak hour (say, 8pm) data:

```
mysql> SELECT … FROM … WHERE … \
        stamp_inserted LIKE '2010-02-% 20:00:00' \
       …
```

Post-processing and reporting (cont.d)

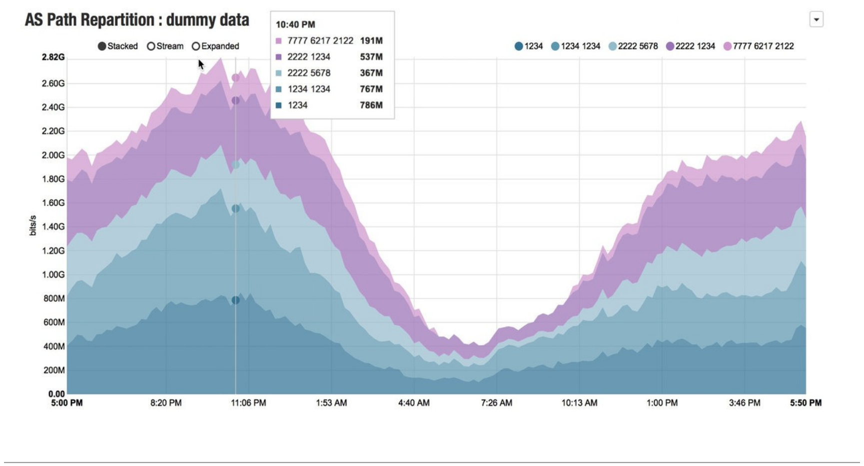
- Traffic breakdown, ie. top N grouping BGP peers of the same kind (ie. peers, customers, transit):

```
mysql> SELECT … FROM … WHERE … \
       local_pref = <<peer | customer | IP transit> pref> \
       …
```

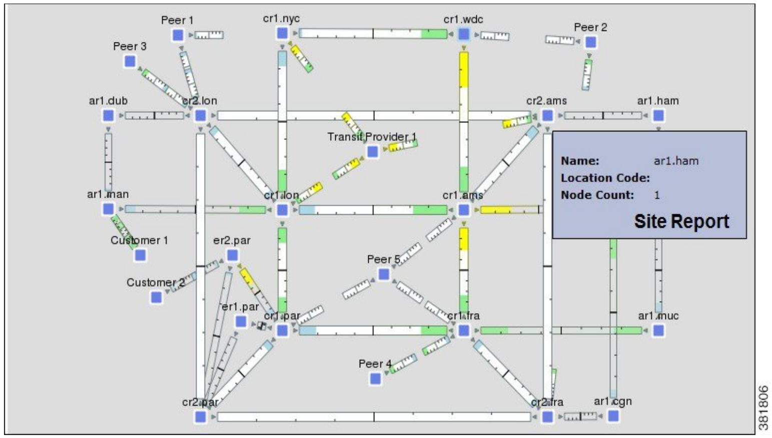
- Traffic matrix (or a subset of it):

```
mysql> SELECT peer_ip_src, peer_ip_dst, bytes, stamp_inserted \
       FROM acct_bgp \
       WHERE [ peer_ip_src = <location A> AND \
             peer_ip_dst = <location Z> AND \ ]
             stamp_inserted = < today | last hour | last 5 mins > \
       GROUP BY peer_ip_src, peer_ip_dst
```

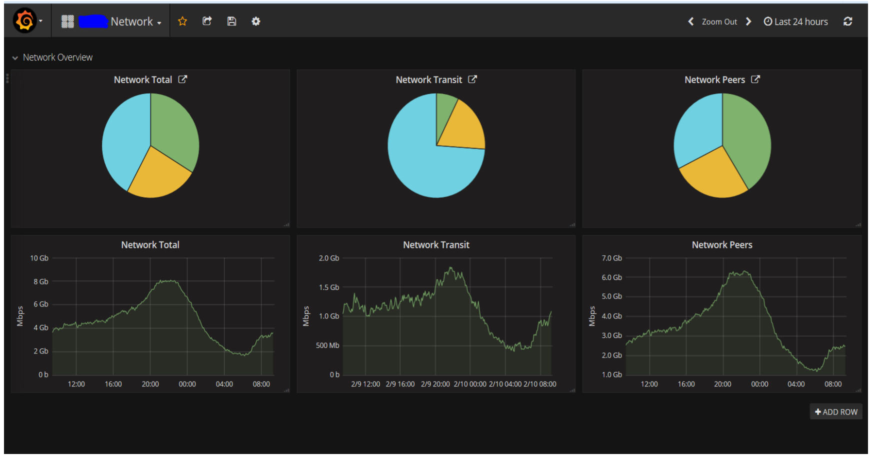
UI



Credits to Dailymotion



Credits to Cisco Systems



Credits to Catalin Petrescu (cpmarvin)

Telemetry data correction

- Telemetry data may get imprecise (ie. due to sampling)
- Use interface stats as gold standard
- Mold telemetry data .. to match interface stats:
  - Builds on Traffic Matrix estimation methods:
    - Tutorial: Best Practices for Determining the Traffic Matrix in IP Networks, NANOG 43
  - Adds telemetry data to linear system to solve
  - Solve system such that there is strict conformance with link stat values, with other measurements matched as best possible

Briefly on scalability

- A single collector might not fit it all:
  - Memory: can’t store all BGP full routing tables
  - CPU: can’t cope with the pace of telemetry export
- Divide-et-impera approach is valid:
  - Assign PEs (both telemetry and BGP) to collectors
  - If natively supported DB:
    - Assign collectors to DB nodes
    - Cluster the DB
  - If not-natively supported DB:
    - Assign collectors to message brokers
    - Cluster the messaging infrastructure

Briefly on scalability (cont.d)

- Intuitively, the matrix can become big:
  - Can be reduced by excluding entities negligible to the specific scenario:
    - Keep smaller routers out of the equation
    - Filter out specific (class of) customers
    - Focus on downstream if CDN, upstream if ISP
    - Sample or put thresholds on traffic relevance

Further information about pmacct

- https://github.com/pmacct/pmacct
  - Official GitHub repository, where star and watch us ☺
- http://www.pmacct.net/lucente_pmacct_uknof14.pdf
  - More about coupling telemetry and BGP
- http://ripe61.ripe.net/presentations/156-ripe61-bcp-planning-and-te.pdf
  - More about traffic matrices, capacity planning & TE
- https://github.com/pmacct/pmacct/wiki/
  - Wiki: docs, implementation notes, ecosystem, etc.

You all invited to the pmacct BoF
AfPIF 2017 Day #3 (Thu) @ 11:00
Plenary room

Building traffic matrices to support peering decisions

Thanks! Questions?

Paolo Lucente <paolo@pmacct.net>

http://www.pmacct.net/ | https://github.com/pmacct/pmacct

AfPIF 2017, Abidjan – Aug 2017